LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-414117

Visualizing LID AR in Google Earth

M. Isenburg, J. Shewchuk

June 23, 2009

Geoinformatics 2009
Faifax, VA, United States
August 12, 2009 through August 14, 2009

Disclaimer

Visualizing LIDAR in Google Earth

Martin Isenburg
Center for Advanced Scientific Computation
Lawrence Livermore National Laboratory

Jonathan Shewchuk
Computer Science Department
University of California at Berkeley

***Abstract*—We describe a set of streaming tools that allow quick visualization of large amounts of LIDAR data in Google Earth. We have found these tools useful to rapidly understand and verify the coverage of acquired data or to easily present this data to a wider audience within a popular geospatial context.**

Starting from raw LIDAR files we use a streaming geometry processing pipeline to turn millions or even billions of LIDAR points into tiled and georeferenced KML files. The generated KML files can either contain elevation contours or image overlays showing color-coded elevation or hillside-shaded terrain.

For Gilmer county, for example, we create a 10 by 8 tiling of 2000 meter tiles containing 10 meter contours in KML format. We do this in only 20 minutes using less than 100 MB of main memory and no temporary disk space on a houshold laptop starting from 357 LAS files that contain a total of 3GB of data which represents a total of 156 million LIDAR points.

All the tools described in this report including the full source code are documented and available from the author's website.

I. Introduction

Modern air-borne laser-range scanning technology (LIDAR) allows collecting elevation samples for large areas. Low flying aircraft shoot up to 100,000 laser pulses per second onto the earth's surface taking measurements at resolutions of one point per squaremeter and an accuracy of around 10 cm. LIDAR data is used in numerous applications: to assess flood hazards, plan solar and wind installations, perform tree inventories, aid in power grid construction, etc. The amount of LIDAR data collected poses a significant computational challenge as not just million but billions of points need to be processed.

One common operation after obtaining LIDAR data is to generate derivatives such as elevation contours or digital elevation maps. These may be the final product or they may just be created to inspect coverage and verify correctness of the data through cross-validation with an existing geospatial context. However, processing of billions of points with conventional tools (such as ArcGIS) requires breaking the data into numerous small pieces. That this can be slow and cumbersome became evident in the first large-scale LIDAR campaign.

Under the NC Floodplain Mapping program, begun after 1999 Hurricane Floyd, all of North Carolina has been flown with LIDAR to map the flood plains to assess risks, set insurance premiums, and create disaster plans. The Raleigh News & Observer reported in March 2002 that "the effort to update flood-plain maps across the state is running years behind original projections, largely because the work is more complex than first imagined... Officials had hoped to finish the job by this year, but now expect to complete it by 2007" [1]. As of today the task has not been completed.

In this paper we describe two streaming geometry processing pipelines that are useful to generate elevation contours and digital elevation maps from massive amounts of LIDAR points for subsequent visualization in Google Earth. Both processing pipelines are based on our streaming Delaunay triangulator that can triangulate billions points on a "household laptop" [2] using very little memory. This method performs all computations in a small memory buffer and outputs results as soon as possible to make room for more points to stream in. The interleaving of I/O and computation allows this software to triangulate much larger LIDAR point sets at much higher speeds than current commercial products.

Fig. 1. Elevation contours derived from bare-earth LIDAR data displayed over Google Earth terrain of Gilmer county, WV (left). The hillside-shaded digital elevation maps derived from first return LIDAR data align perfectly with the Google Earth imagery of Fitchburg Muncipal Airport, MA (right).

We have shown that the triangles that are streaming out of our Delaunay triangulator can be directly processed, either by immediately rasterizing them onto a digital elevation map [3] or by immediately extracting elevation contours from them [4]. This allows implementations to work *out-of-core* making it possible to process input data and to produce output data that are much larger than the available main memory. We have made these tools available on our website and the feedback from the LIDAR processing community has been positive—"your tools have been far faster and much more reliable [than ArcGIS]" to quote one typical testimonial.

In this paper we detail an extension of our streaming pipelines [3] and [4] to output tilings of georeferenced elevation contours and digital elevation maps in KML format so they can directly be ingested into Google Earth for immediate visualization within a popular geospatial context.

II. Overview

The beginning of our streaming pipeline is always the same: The first module spatially finalizes the point stream. Our

sp□nalize software adds □nalization tags to a stream of raw LIDAR points that mark the moment in which the speci□ed cell of a recursively divided bounding box has "seen" all points of the stream that fall into it. The second module Delaunay triangulates the point stream while taking advantage of the □nalization tags. Our *sp2delaunay* software uses the tags to certify triangles as Delaunay, output them in a streaming mesh format, and free up or rather re-use the memory they were occuping. The exact workings of *sp□nalize* and *spdelaunay2d* are detailed in an earlier paper [2]. The rest of the streaming pipeline is different depending on whether we want to extract elevation contours or rasterize digital elevation maps.

a) Extracting elevation contours: is done with a third module that processes the triangles as they stream out of the triangulator [4]. Our *tin2iso* software checks the elevations of each of the triangle's three vertices and outputs the requested contour lines in a streaming line format. The fourth and □fth modules are called *slclean* and *slsimp* and—as their names suggests—clean and simplify the produced contours as they are getting produced. The cleaning is done by removing tiny closed contours that usually correspond to unwanted details in the LIDAR data. The simpli□cation is done by joining two adjacent line segments into one whenever the area they enclose falls under a user-de□ned threshold. The □nal module is called *sl2sl* and it reads the cleaned and simpli□ed streaming lines and tiles them into KML □les for display in Google Earth using the georeferencing information the user provides.

b) Rasterizing digital elevation maps: is done with a third module that rasterizes the triangles as they stream out of the triangulator [3]. If the user requested hillside-shaded terrain our *tin2dem* software computes per-vertex normals while the triangles stream in that are used to generate the correct shading with respect to the speci□ed light direction. The rasterizer writes the generated rasters to temporary □les based on the tiling that was requested by the user. In a □nal pass over the temporary □les it produces PNG, JPG, or TIF □les together with KML □les that geo-reference the imagery in Google Earth based on the projection information provided by the user.

III. Streaming Meshes, Points, and Lines

Keeping points, triangles, and lines in streaming formats is key to the ef□ciency of our approach. It allows using streaming algorithms for each processing task that can operate on data much larger than the main memory. It also allow using simple commandline piping to stream data from one module to the next. The operating system automatically allocates the most processing time to the busiest modules and—when available—can immediately take advantages of multiple cores. It also makes is possible to quickly recon□gure the processing pipeline when more ef□cient or new modules become available or when user demands are changing. For example, we can plug our streaming simpli□cation software [5] between the triangulator *spdelaunay2d* and the elevation contour extractor *tin2iso* to coarsen the terrain before extracting the contours. This can generate much higher quality simpli□cations of elevation contours that are guaranteed to be self-intersection free. This comes, however, at the expense of slower performance as terrain simpli□cation is a fairly compute intensive operation.

Performing streaming operations on point data requires spatial □nalization: space is partitioned into regions, and a region is □nalized after the last point in the region appears in the stream. In our pipeline we need spatially □nalized points as input to our streaming Delaunay triangulator. In the ideal case the input format from which we start processing already contains some for of spatial □nalization. This would be true if we were to request the points from a spatial database. However, a typical use scenario for our tools starts with a list of LIDAR □les in LAS format. Here we need to compute spatial □nalization with two additional passes over the points.

Isenburg and Lindstrom [5] describe a streaming format for meshes: vertices and triangles are intermix ed, along with vertex □nalization tags that indicate when all triangles referencing this vertex have already appeared in the stream. This topological □nalization of mesh vertices allows modules to derive when, for example, the one-ring neighborhood around a vertex has completely appeared in the stream such that tasks that need data from such neighborhoods (e.g. computing vertex normals) can be performed immediately. Finalization of vertex v tells the application that it can complete all computations that were waiting for vs topology, output partial results, and safely free any data structures that are no longer needed.

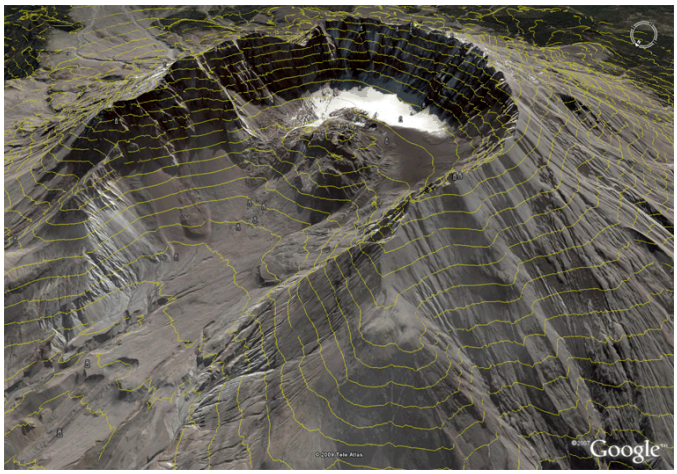

Fig. 2. Elevation contours in 50 meter intervals for Mount St. Helen derived from bare-earth LIDAR data displayed over Google Earth terrain.

In our pipeline we produce triangulated terrain in a topologically □nalized streaming mesh format as output of our streaming Delaunay triangulator so that we can, for example, on-the-□y compute per-vertex normals that are needed for hillshading shading the terrain. We also use this topological □nalization in the format to extract elevation contours and output them in a streaming line format. Similar to the streaming mesh format, the streaming line format interleaves vertices and line segments and includes tags that □nalize vertices that are have "seen" all their line segments. Having topologically □nalized lines can in turn be exploited for streaming topological cleanup that removes connected components smaller than some

threshold. We only need to buffer connected line segments that have active vertices (i.e. vertices that are not yet □nalized) and are below the threshold. Whenever all vertices of a component are □nalized before the component reaches the threshold then the entire component can be discarded. Whenever a component reaches the threshold while still having active vertices we can output it. Any future line segments that connect to this component can then immediately be output as well.

IV. Visualizing Elevation Contours

We now describe in detail our streaming pipeline for extracting isocontours from LID AR data and storing them as georeferenced KML □les for immediate visualization in Google Earth. We do this using 156 million bare-earth LID AR points from Gilmer county, West Virginia that are stored in 357 LAS □les that total 3GB as an example. In particular, we describe the commandline that creates a 10 by 8 tiling of 2000 meter tiles containing 10 meter contours in KML format. The entire process runs in only 20 minutes using less than 100 MB of main memory and no temporary disk space on a Dell Inspiron 6000 laptop. The raw data and the commandline tools are available can be found on the Web [6] for anyone wanting to validate our results or wishing to experiment with their own data. The resulting KML □les for Gilmer county as well as several other examples are available there as well.

```
sp□nalize -i gilmer.□les -lof -ilas -level 8 -ospb |
    spdelaunay2d -ispb -osmb |
    tin2iso -ismb -range 200 450 10 -oslb |
    slclean -islb -oslb -length 5 |
    slsimp -islb -oslb -area 0.7 |
    sl2sl -islb -okml -utm 17S -ellipsoid 23
        -tiling _nllsxy gilmer 500000 4302000 2000 10 8
```

Fig. 3. The command line pipe that creates a 10 by 8 tiling of 2000 meter tiles containing 10 meter contours in KML format from 156 million bare-earth LID AR points of Gilmer county that total 3GB stored in 357 LAS □les.

The □rst module **sp□nalize.exe** reads the text □le 'gilmer.□les' *-i gilmer.□les* that contains a list to the locations of 357 □les *-lof* that contain the LID AR points in LAS format *-ilas*. The module □nalizes the points from these 357 □les onto a 256x256 grid *-level 8* and outputs them in a streaming binary point format *-ospb* to the second module.

The second module **spdelaunay2d.exe** reads the □nalized points in streaming binary format *-ispb* as they are produced by the □nalizer and Delaunay triangulates them with our streaming algorithm [2]. The module immediately starts to output the triangulation in a binary streaming mesh format *-osmb* and pipes it to the third module.

The third module **tin2iso.exe** reads the triangulation from the binary streaming mesh format *-ismb* as it is produced by the streaming triangulator, extracts elevation contours every 10 meters between 200 and 450 *-range 200 450 10* and outputs the resulting elevation contour lines in a binary streaming line format *-oslb* to the fourth module.

The fourth module **slclean.exe** reads the contours in binary streaming line format *-islb* as they are produced by the extractor, discards all contours that are shorter than 5 meter *-length 5*, and starts outputting others as soon as their length is determined as being above the cutoff of 5 meters in a binary streaming line format *-oslb* to the □fth module.

The □fth module **slsimp.exe** reads the contours in binary streaming line format *-islb* as they are output by the cleaner, removes all 'bumps' (i.e. pairs of two subsequent line segments) that are less than 0.7 square meter in area *-area 0.7*, and outputs the simpli□ed elevation contours in a binary streaming line format *-oslb* to the sixth module.

The sixth module **sl2sl.exe** reads the contours in binary streaming line format *-islb* as output by the simpli□er and tiles them into $x = 0..9$ by $y = 0..7$ separate □les called 'gilmer_0x_0y.kml' with $(500000, 4302000)$ being the lower left corner of the tiling and with each tile being 2000 meters long and wide *-tiling _nllsxy gilmer 500000 4302000 2000 10 8*; each tile is stored in Google's KML format *-okml*. Because KML uses longitude and latitude in degrees and elevation in meter we to convert the LID AR data to a correctly georeferenced representation. Since the LID AR data was in UTM format and we need to specify the UTM zone *-utm 17S* and the ellipsoid WGS-84 *-ellipsoid 23*.

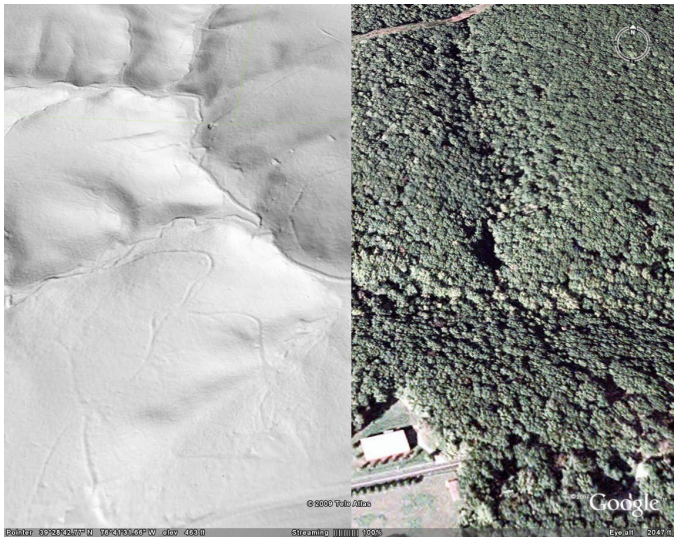

Fig. 4. Hillside-shaded digital elevation maps of Baisman Run in Baltimore County derived from bare-earth LID AR data displayed side by side with Google Earth terrain imagery. It is as if one can “see” through the canopy.

Because our elevation data is much more precise than the terrain data used by Google Earth our elevation contours do not align perfectly with the terrain (see Fig. 1). The lower resolution ravines in Google Earth are less deep, so our isolines run below the terrain, while the ridges in GE are less tall, so our isolines □oat above the terrain. When we started this project we noticed that the elevation of Google Earth terrain for Mount Saint Helens (see Fig. 2) did not at all match the contours around the lava dome that is inside the crater. That was because this lava dome had grown by about 330 feet in 2004 and Google Earth had still been using elevation data from the Shuttle Radar Topography Mission of 2000. In the

meantime their elevations have been updated with newer data of higher resolution and our contours align well.

V. Creating Digital Elevation Maps

We now describe our streaming pipeline for deriving either hillside-shaded or elevation color-coded digital elevation maps from LIDAR data and storing them as a tiling of georeferenced KML files for immediate visualization in Google Earth.

```
sp□nalize -i gilmer.□les -lof -ilas -level 8 -ospb |
    spdelaunay2d -ispb -osmb |
    tin2dem -ismb -opng -zone 17S -ellipse 23 -step 5 |
            -tiling _ns gilmer 2500 -ll 500000 4302000
```

Fig. 5. The command line pipe that creates a tiling of 2500 meter tiles containing hillside-shaded DEM in PNG format together with a KML file that correctly georeferences the tiled imagery within Google Earth.

The first module **sp□nalize.exe** and the second module **spdelaunay2d.exe** operate exactly as described in the last section and pipe triangulated LIDAR points in binary streaming mesh format to the third module.

The third module **tin2dem.exe** reads the triangulation from the binary streaming mesh format *-ismb*, and rasters it with hillside shading into PNG files *-opng* with a step size of 5 meter per pixel *-step 5* creating a tiling with 2500 meter tiles (giving us image tiles of 500 by 500 pixels) that is called 'gilmer' *-tiling _ns gilmer 2500* starting at the lower left corner with coordinates (500000,4302000) *-ll 500000 4302000*.

VI. Discussion

Our streaming geometry processing pipelines are a fast way to derive elevation contours and digital elevation maps from large amounts of LIDAR data. Because the computation is streaming and data-driven our method scales to gigantic inputs and outputs: we can process gigabytes of data equaling billions of LIDAR points on a household laptop [3]. We have described how to extend these processing pipelines to directly turn LIDAR points into geo-referenced Google Earth tilings for immediate preview or distribution to a wider audience.

The ability to view LIDAR data within the powerful visualization platform provided by Google Earth allows for interesting insights, as the data suddenly presents itself surrounded by a geospatial context that is annotated by a multitude of users and providers. For example, we had been experimenting with a data file called "Serpent_Mound.las" for quite some time, but it was not until we viewed the corresponding DEM within Google Earth that we actually noticed the serpent. Several geospatial tags were pointing out the presence of "The Great Serpent Mound", a 1,330-foot-long, three-foot-high prehistoric effigy mound located on a plateau of the Serpent Mound crater along Ohio Brush Creek in Adams County, Ohio. The serpent is the largest effigy earthwork in the world and it appears nowhere as crisp and beautiful as in the hillside-shaded DEM that we derived from bare-earth LIDAR (see Fig. 6).

Another interesting variation is to use Google Maps instead of Google Earth for displaying the derived elevation contours. In Fig. 7 we show an example of downtown Toronto for

Fig. 6. Our hillside-shaded digital elevation map clearly exposes "The Great Serpent Mound" that is hidden under the canopy in the Google Earth imagery.

contours derived from first return LIDAR data. In this case the elevation contours are giving us exact and georeferenced footprints of the buildings from that part of the map.

Fig. 7. The elevation contours derived from first return LIDAR data of downtown Toronto are displayed as an overlay in Google Maps. This puts the footprint of various famous buildings at their exact locations on the map.

The authors would like to thank everybody who has downloaded our tools and sent suggestions and/or bug reports.

References

[1] M. Quillin, "Flood plain maps better, but late – years late," March 11 2002, raleigh News & Observer.

[2] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink, "Streaming computation of Delaunay triangulations," in *Proceedings of SIGGRAPH'06*, 2006, pp. 1049–1056.

[3] M. Isenburg, Y. Liu, J. Shewchuk, J. Snoeyink, and T. Thirion, "Generating raster DEM from mass points via TIN streaming," in *GIScience'06 Conference Proceedings*, 2006, pp. 186–198.

[4] M. Isenburg, Y. Liu, and J. Snoeyink, "Streaming extraction of elevation contours from lidar points," in *manuscript*, 2006.

[5] M. Isenburg and P. Lindstrom, "Streaming meshes", in *Visualization'05 Proceedings*, 2005, pp. 231–238.

[6] M. Isenburg and J. Shewchuk, "Visualizing lidar in google earth," in *http://www.cs.unc.edu/~isenburg/googleearth/*.

This work performed under the auspices of the U.S. DOE by LLNL under Contract DE-AC52-07NA27344.